INFORMATION AND COMPUTATION **130**, 131–150 (1996)
ARTICLE NO. 0086

Parametric Timing Analysis for

Farn Wang

Institute of Information Science, Academia Sinica, Nankang, Taipei, Taiwan 115, Republic of China
E-mail: farn@iis.sinica.edu.tw

We extend a TCTL model-checking problem to a parametric timing analysis problem for real-time systems and develop new techniques for solving it. The algorithm we present here accepts timed transition system descriptions and parametric TCTL formulas with timing parameter variables of unknown sizes and can give back general linear equations of timing parameter variables whose solutions make the systems work.

1. INTRODUCTION

The verification problems of reactive systems have traditionally been modeled as decision problems which use Boolean values as answers [ACD90, AFH91, AH90, EMSS90, HLP90, HNSY92, Lewis90, WME93a, WME93b]. One such prominent example is the model-checking problem of CTL [CE81, CES86] which, given a transition system description A and a CTL formula ϕ, asks if A is a model of ϕ. Alur *et al.* have extended the CTL model-checking problem to real-time systems with dense time-domain by equipping transition systems with clocks that can be reset [ACD90].

The framework of decision problems is not all that natural to system designers. In seeking a working design among numerous plausible choices, people prefer more informative answers. Moreover, the commonly adopted frameworks in temporal logics, like that of TCTL model-checking, usually require overly detailed specifications of system state configurations. For example, we must know how many binary variables (or variables encodable in predefined number of bits) there are before the analysis can proceed. This kind of restriction usually leaves users in repetitive trial-and-error cycles to select a parameter valuation.

* A preliminary version of this paper appears in the proceedings of the 10th IEEE Symposium on Logic in Computer Science, San Diego, CA, June 1995.

0890-5401/96 $18.00

As Alur *et al.* have observed [AHV93],

> Indeed, when studying the literature on real-time protocols, one sees that the desired timing properties for protocols are almost invariably parametric.

However, it is easy to show that if we let users have parameters of unknown sizes in specifications and formulas, the verification problems can easily become undecidable. The contribution of this paper is that we extend a TCTL model-checking problem to a *parametric timing analysis problem* with unknown timing parameters and provide an elementary complexity algorithm for the general solution condition of the problem. Specifically, we define *Parametric TCTL* (*PTCTL*), which allows formulas like $\exists \phi_1 \mathcal{U}_{>\sigma} \phi_2$, where σ is an unknown nonnegative integer variable. Given an automaton A with resettable clocks and a PTCTL formula ϕ, we use $PTA(A, \phi)$ to denote the parametric timing analysis problem instance which asks for the general condition on a parameter valuation, if there is any, of unknown timing variables in ϕ that makes A a model of ϕ. We show that such conditions are always expressible as sets of linear equations. The design of the algorithm requires new insight and interesting techniques in analyzing timed transition systems which seem to have great potential in helping attacking other real-time system analysis problems. Especially in our framework, the region graph construction is independent of the timing constants used in the specification formula. This should help in containing the exponential growth of space requirement for verification tasks.

1.1. Related Work

In the earliest development [CE81, CES86], people used finite-state automata to describe system behavior and check to see if it satisfied specifications given in branching-time temporal logic CTL. Such a framework is usually called *model-checking*. A *CTL* (*Computation Tree Logic*) formula is composed of binary propositions (p, q, ...), Boolean operators ($\neg$, $\vee$, $\wedge$), and branching-time modal operators ($\exists\mathcal{U}$, $\exists\bigcirc$, $\forall\mathcal{U}$, $\forall\bigcirc$). $\exists$ means "*there exists*" a computation. $\forall$ means "*for all*" computations. $\mathcal{U}$ means something is true "until" something else is true. $\bigcirc$ means "*next state*." For example, $\exists p \mathcal{U} q$ says there exists a computation along which p is true until q is true. Since there is no notion of real-time (clock time), only ordering among events is considered.

The following shorthand notations are generally accepted, besides the usual ones in Boolean algebra. $\exists\Diamond\phi_1$ is for $\exists true\, \mathcal{U} \phi_1$; $\forall\Box\phi_1$ for $\neg\exists\Diamond\neg\phi_1$; $\forall\Diamond\phi_1$ for $\forall true\, \mathcal{U} \phi_1$; and $\exists\Box\phi_1$ for $\neg\forall\Diamond\neg\phi_1$. Intuitively, $\Diamond$ means "*eventually*" while $\Box$ means "*henceforth*."

The ground-breaking paper in [ACD90] presents a dense-time system model-checking algorithm. The system behavior is described by finite-state automaton extended with dense-time clocks. In the automata, we can compare the difference of two clock readings with timing constants to trigger transitions and reset them during the transitions. The specification is given in *TCTL* (*Timed CTL*) which extends CTL with timing constraints like deadline, earliest starting time, and exact time. A typical real-time automaton, which has two clocks, x, y, and four meta-states, q_0,

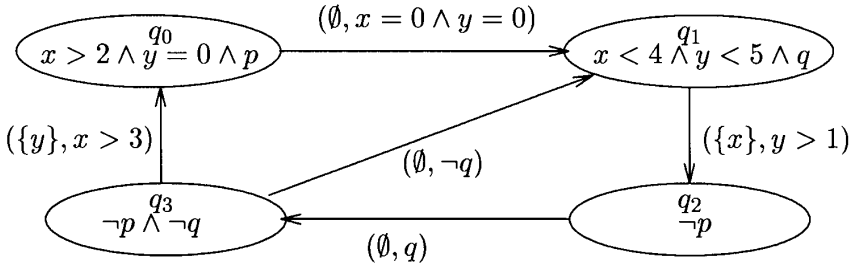

FIG. 1. A clock automaton.

q_1, q_2, q_3, is shown in Fig. 1. At each meta-state, we have a condition to be maintained. For example, in meta-state q_0, $x > 2 \wedge y = 0 \wedge p$ must always be true. By each transition, we have a pair whose first component is the set of clocks to be reset and whose second component is the triggering condition. The automaton may go from meta-state q_1 to q_2 and reset x's reading to zero when the reading of y is greater than one. A typical TCTL formula is $\exists \Diamond_{\leqslant 5} p$ which says that in some computation that p will sometimes be true after 5 time units. Because of the dense-time nature, usually no next-state operator is adopted. One innovation in [ACD90] involves a partitioning technique which divides the infinite state space of such dense-time systems into finite number of behaviorally equivalent *regions*.

The framework of PTCTL and timed automata is chosen partly for convenience of algorithm presentation. It is possible to restate our result in the general framework of parametric real-time reasoning given in [AHV93] which focussed on the emptiness problems of parametric timed automata with various restrictions.

In [CY92], the problems of deciding the earliest and latest times a target state can appear in the computation of a timed automaton were discussed. However, we do not know of any previous work on deriving the general characteristic formulas of solution parameter valuations.

1.2. Outline of the Rest of the Paper

In Section 2, we start the presentation by giving the intuition behind the work. In Section 3, we introduce clock automata as our transition system description language. Section 4 defines the syntax and semantics of PTCTL and the problem of parametric timing analysis; especially, the semantics are straightforwardly defined on clock automata for convenience. Section 5 describes our observations and algorithms for solving the problem. Intuitive observations behind our labeling algorithm and intrinsic function solution algorithm are given at the beginning of Sections 5.2 and 5.5, respectively. Section 6 is the conclusion.

Notationally, we let $\mathcal{N}$ be the set of nonnegative integers and $\mathcal{R}^+$ the set of nonnegative real numbers.

2. INTUITIVE BASIS OF OUR APPROACH

In this section, we try to give a small region graph example to show the intuition behind the work. Suppose we have the region graph depicted in Fig. 2 with a single clock variable x and we want to check what the condition on parameter variable

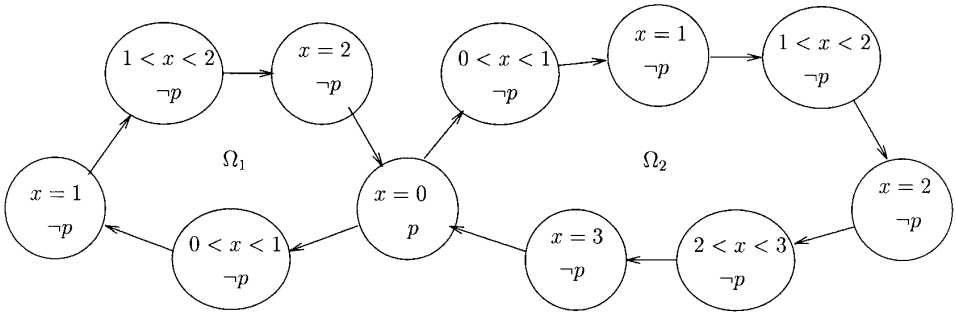

FIG. 2. A region graph.

σ is in order to make $\exists \Diamond_{=\sigma} p$ true at the center p-state. There are two cycles in the region graph, Ω_1 and Ω_2, with cycle computation times 2 and 3, respectively. It is obvious that 0 is a solution to σ. Also, any multiples of 2 and 3 are solutions. In fact, by looping Ω_1 and Ω_2 any number of times, we will still come back to the center p-state. This suggests that the solution to σ will be an all linear combinations of 2 and 3. In Section 5, we shall formalize this intuition by considering all cases. It turns out that we can construct an initial offset computation time such that after the initial offset computation time, any extra computation time equal to a multiple of the gcd (*greatest common divisor*) of all the "positive" cycle times results in a solution.

In other words, the solution computation times eventually stablize to some periodic pattern. This observation not only leads to the parametric timing analysis algorithm for such region graphs but also makes the region graph construction independent of the specification formula.

In Section 5.2, we shall derive the *initial offset computation time* with the help of lemmas 2 and 3 and thereafter prove that the intuition is correct.

3. CLOCK AUTOMATA

Our system models are described by *clock automata* (*CA*) which are dense-time automata similar to the timed graphs used in [ACD90] and timed safety automata in [HNSY92]. A CA has a set of clocks whose readings are nonnegative real numbers that increase at a uniform rate and can be compared with integer constants. Each clock can be reset to zero independently. For example, the CA in Fig. 1 has four meta-states and two clock variables (see Section 1.1 for an explanation of its operation).

Given a set P of atomic propositions and a set C of clocks, the syntax of a *state predicate* η of P and C is defined as

$$\eta ::= \mathit{false} \mid p \mid x \sim c \mid \eta_1 \rightarrow \eta_2.$$

Here p is an atomic proposition in P, x is a clock in C, c is a nonnegative integer constant, and $\sim$ stands for one of the binary relations $<$, $\leqslant$, $=$, $\geqslant$, $>$. Let B_P^C

be the set of all state predicates of P and C. We shall also adopt $\neg\eta_1$, *true*, $\eta_1 \vee \eta_2$, $\eta_1 \wedge \eta_2$ as shorthand notations for $\eta_1 \rightarrow$ *false*, $\neg$*false*, $(\neg\eta_1) \rightarrow \eta_2$, $\neg(\eta_1 \rightarrow \neg\eta_2)$, respectively.

DEFINITION 1. *Clock Automata.* A clock automaton (CA) is a tuple $(Q, q_0, P, C, \chi, E, \pi, \tau)$ with the following restrictions.

- Q is a finite set of meta-states.
- $q_0 \in Q$ is the initial meta-state.
- P is a set of atomic propositions.
- C is a set of clocks.
- $\chi: Q \mapsto B_P^C$ is a function that labels each meta-state with a condition that is true in that meta-state.
- $E \subseteq Q \times Q$ is the set of transitions.
- $\pi: E \mapsto 2^C$ defines the set of clocks to be reset during each transition.
- $\tau: E \mapsto B_P^C$ defines the transition triggering conditions.

The CA starts at meta-state q_0. The transitions of the CA are triggered by state predicates. During a transition from q_i to q_j, for each $x \in \pi(q_i, q_j)$, the reading of x will be reset to zero.

A *state* s of CA $A = (Q, q_0, P, C, \chi, E, \pi, \tau)$ is a mapping from $P \cup C$ to $\{$*true*, *false*$\} \cup \mathscr{R}^+$ such that for each $p \in P$, $s(p) \in \{$*true*, *false*$\}$, and for each $x \in C$, $s(x) \in \mathscr{R}^+$. A state predicate η can be satisfied in a state s, written as $s \models \eta$. The relation of satisfaction of state predicates by states is defined by the following four rules:

- $s \not\models$ *false*;
- $s \models p$ iff $s(p) =$ *true*;
- $s \models x \sim c$ iff $s(x) \sim c$; and
- $s \models \eta_1 \rightarrow \eta_2$ iff $s \models \eta_1$ implies $s \models \eta_2$.

A CA $A = (Q, q_0, P, C, \chi, E, \pi, \tau)$ is *unambiguous* iff for all states s, there is at most one $q \in Q$ such that $s \models \chi(q)$. Ambiguous CA's can be made unambiguous by incorporating meta-state names as propositional conjuncts in the conjunctive normal forms of corresponding $\chi(\ )$ mapping values. From now on, we shall only talk about unambiguous CA's. When we say a CA, we mean an unambiguous CA. Given a CA $A = (Q, q_0, P, C, \chi, E, \pi, \tau)$ and a state s, we shall let s^Q be the meta-state in Q such that $s \models \chi(s^Q)$. If there is no meta-state $q \in Q$ such that $s \models \chi(q)$, then we shall say s^Q is undefined.

In a CA, there are three causes of state changes: (1) meta-state transitions, (2) proposition value changes in the same meta-state, and (3) time passage in the same meta-state. The second case can be emulated by the first one by requiring at each meta-state, there is a self-loop transition with tautology triggering condition. So

from now on, we shall always ignore the second case by assuming that for each meta-state, there is such a tautology-triggered self-loop transition.

Given two states s, s', we say there is a meta-state transition from s to s' in A, in symbols $s \to s'$, iff

- s^Q, s'^Q are both defined,
- $(s^Q, s'^Q) \in E$,
- $s \models \tau(s^Q, s'^Q)$, and
- $\forall x \in C((x \in \pi(s^Q, s'^Q) \to s'(x) = 0) \wedge (x \notin \pi(s^Q, s'^Q) \to s'(x) = s(x)))$.

Also given a state s and a $\delta \in \mathscr{R}^+$, we let $s + \delta$ be the state that agrees with s in every aspect except for all $x \in C$, $s(x) + \delta = (s + \delta)(x)$. Given a state s of a CA $A = (Q, q_0, P, C, \chi, E, \pi, \tau)$, a computation of A starting at s is called an s-run, which can be represented by an infinite sequence $((s_1, t_1), (s_2, t_2), \ldots)$ such that

- $s = s_1$; and
- for each $t \in \mathscr{R}^+$, there is an $i \in \mathscr{N}$ such that $t_i \geqslant t$; and
- for each integer $i \geqslant 1$, s_i^Q is defined and for each real $0 \leqslant \delta \leqslant t_{i+1} - t_i$, $s_i + \delta \models \chi(s_i^Q)$; and
- for each $i \geqslant 1$, A goes from s_i to s_{i+1} because of
  - — meta-state transition, i.e., $t_i = t_{i+1} \wedge s_i \to s_{i+1}$, or
  - — time passage, i.e., $t_i < t_{i+1} \wedge s_i + t_{i+1} - t_i = s_{i+1}$.

4. PARAMETRIC TCTL AND PARAMETRIC TIMING ANALYSIS PROBLEM

We extend TCTL [ACD90] to Parametric TCTL to specify timing properties in our parametric timing analysis problem. Each formula ϕ in PTCTL is accompanied by a set T_ϕ of parameter variables which are character strings representing unspecified timing constants. The syntax of a PTCTL formula ϕ, used for analyzing models described by a CA $(Q, q_0, P, C, \chi, E, \pi, \tau)$, is defined by

$$\phi ::= \mathit{false} \mid p \mid \phi_1 \to \phi_2 \mid \exists \phi_1 \mathscr{U}_{\sim\theta} \phi_2 \mid \forall \phi_1 \mathscr{U}_{\sim\theta} \phi_2.$$

Here p is an atomic proposition name in P. θ is either an integer constant in $\mathscr{N}$ or a parameter variable in T_ϕ. We shall define abbreviations $\neg\phi_1$ for $(\phi_1 \to \mathit{false})$, *true* for $\neg \mathit{false}$, $\phi_1 \vee \phi_2$ for $(\neg\phi_1) \to \phi_2$, $\phi_1 \wedge \phi_2$ for $\neg(\phi_1 \to \neg\phi_2)$, $\exists \Diamond_{\sim\theta} \phi_1$ for $\exists \mathit{true}\, \mathscr{U}_{\sim\theta} \phi_1$, $\forall \Box_{\sim\theta} \phi_1$ for $\neg \exists \Diamond_{\sim\theta} \neg\phi_1$, $\forall \Diamond_{\sim\theta} \phi_1$ for $\forall \mathit{true}\, \mathscr{U}_{\sim\theta} \phi_1$, and $\exists \Box_{\sim\theta} \phi_1$ for $\neg \forall \Diamond_{\sim\theta} \neg\phi_1$.

A parameter *valuation*, say $\mathscr{I}$, for T_ϕ is a mapping from $\mathscr{N} \cup T_\phi$ to $\mathscr{N}$ such that for all $c \in \mathscr{N}$, $\mathscr{I}(c) = c$. With different parameter valuations, a PTCTL formula may impose different timing requirements. Given a PTCTL formula ϕ and a parameter valuation $\mathscr{I}$ for T_ϕ, we shall let $\phi^{\mathscr{I}}$ be the TCTL formula [ACD90] obtained from ϕ by replacing every occurrence of σ in ϕ by $\mathscr{I}(\sigma)$ for all $\sigma \in T_\phi$.

We now define the semantics of PTCTL with parameter valuation on CA's. We write $s \models \phi^{\mathscr{I}}$ to mean that ϕ with parameter valuation $\mathscr{I}$ is true in A at state s. We define $\models$ inductively as follows.

- $s \not\models \mathit{false}$
- $s \models p^{\mathscr{I}}$ iff $s(p) = \mathit{true}$
- $s \models (\phi_1 \to \phi_2)^{\mathscr{I}}$ iff $s \models \phi_1^{\mathscr{I}}$ implies $s \models \phi_2^{\mathscr{I}}$
- $s \models (\exists \phi_1 \mathscr{U}_{\sim\theta} \phi_2)^{\mathscr{I}}$ iff there are an s-run $= ((s_1, t_1), (s_2, t_2), \ldots)$ in A, an $i \geqslant 1$, and a $\delta \in [0, t_{i+1} - t_i]$, s.t.
  - $t_i + \delta \sim t_1 + \mathscr{I}(\theta)$,
  - $s_i + \delta \models \phi_2^{\mathscr{I}}$,
  - for all $0 \leqslant j < i$ and $\delta' \in [0, t_{j+1} - t_j]$, $s_j + \delta' \models \phi_1^{\mathscr{I}}$, and
  - for all $\delta' \in [0, \delta)$, $s_i + \delta' \models \phi_1^{\mathscr{I}}$.
- $s \models (\forall \phi_1 \mathscr{U}_{\sim\theta} \phi_2)^{\mathscr{I}}$ iff for every s-run $= ((s_1, t_1), (s_2, t_2), \ldots)$ in A, for some $i \geqslant 1$ and $\delta \in [0, t_{i+1} - t_i]$,
  - $t_i + \delta \sim t_1 + \mathscr{I}(\theta)$,
  - $s_i + \delta \models \phi_2^{\mathscr{I}}$,
  - for all $0 \leqslant j < i$ and $\delta' \in [0, t_{j+1} - t_j]$, $s_j + \delta' \models \phi_1^{\mathscr{I}}$, and
  - for all $\delta' \in [0, \delta)$, $s_i + \delta' \models \phi_1^{\mathscr{I}}$.

Given a CA A, a PTCTL formula ϕ, and a parameter valuation $\mathscr{I}$ for T_ϕ, we say A is a *model* of $\phi^{\mathscr{I}}$, written as $A \models \phi^{\mathscr{I}}$, iff $s \models \phi^{\mathscr{I}}$ for all states s such that $s^Q = q_0$. A parameter valuation $\mathscr{I}$ is called a *solution* of $PTA(A, \phi)$ iff $A \models \phi^{\mathscr{I}}$. The *parametric timing analysis problem* instance for A and ϕ, i.e., $PTA(A, \phi)$, is formally defined as the problem of deriving the solution condition on a parameter valuation $\mathscr{I}$, if any, which makes $A \models \phi^{\mathscr{I}}$.

5. PARAMETRIC TIMING ANALYSIS

Typical questions in analyzing our problem instances come in the following way: If we are given two states, s and s', of A, what are the conditions on times of computation from s to s'? We shall use the region graphs defined in [ACD90] as a basis for developing our algorithm. We find that the repetition patterns of path time from s to s' are expressible in terms of linear inequalities with coefficients linear in the gcd's and lcm's of the computation times of positive simple cycles traversable by paths from s to s'. This relationship can then be expressed as conditions on parameter variables in T_ϕ.

A formal definition of our region graphs will be given in Section 5.1. The derivation of the linear inequalities related to the solution conditions will be presented in Section 5.2. Then we apply the results to label the regions with conditions on parameter variables in Section 5.3. The complexity of the labeling algorithm is

discussed in Section 5.4. Finally, we show how to further simplify the linear inequalities to linear equations and we suggest the use of a standard technique for linear equations to derive a solution on the parameter variables in Section 5.5.

5.1. Clock Region Graph

Clock region graphs (*CR-graphs*) are basically the region graphs defined in [ACD90] supplemented with information to support analysis of timing parameters. Specifically, we extend the region graphs with a clock tick indicator κ which is conceptually a clock that gets reset to zero once its reading reaches one. Moreover, we ask that the reading of κ always be between 0 and 1; that is, for every state s, $0 \leqslant s(\kappa) \leqslant 1$.

We let K_A and K_ϕ be the biggest constants used in A and ϕ, respectively. For each $\delta \in \mathscr{R}^+$, we define $fract(\delta)$ as the fractional part of δ; i.e., $fract(\delta) = \delta - \lfloor \delta \rfloor$. We shall slightly modify the concepts of region (equivalence classes of states w.r.t. $PTA(A, \phi)$) and region graph [ACD90] to develop our algorithm.

DEFINITION 2. *K-region*. Given a clock automaton $A = (Q, q_0, P, C, \chi, E, \pi, \tau)$, a nonnegative integer constant K, and two states s, s' of A, $s \cong_{A:K} s'$ (i.e., s and s' are equivalent with respect to A and K) iff the following conditions are met:

- For each $p \in P$, $s(p) = s'(p)$.
- For each $x \in C$, if either $s(x) \leqslant K$ or $s'(x) \leqslant K$, then $\lfloor s(x) \rfloor = \lfloor s'(x) \rfloor$.
- For every x, $y \in C \cup \{0, \kappa\}$, $fract(s(x)) \leqslant fract(s(y))$ iff $fract(s'(x)) \leqslant fract(s'(y))$. (Here we conveniently let $s(0) = s'(0) = 0$.)

When $K = K_A$, we shall write $s \cong_A s'$ instead. When the context of A and K is obvious, we shall write $s \cong s'$ for simplicity. $[s]$ denotes the equivalent class of A's states to which s belongs. We call each equivalent class a *K-region* (or simply a *region* when $K = K_A$).

We now restate a lemma from [ACD90] without proof.

LEMMA 1. *Suppose we are given* $PTA(A, \phi)$, *parameter valuation* $\mathscr{I}$ *for* T_ϕ, *and two states* s, s' *of* A *such that* $s \cong_{A: \max(K_A, K_\phi)} s'$. *Then* $s \models \phi^{\mathscr{I}}$ *iff* $s' \models \phi^{\mathscr{I}}$.

Now the clock region graphs are defined as follows:

DEFINITION 3. *CR-graph*. The CR-graph for $PTA(A, \phi)$, denoted as G_A, with $A = (Q, q_0, P, C, \chi, E, \pi, \tau)$, is a directed graph (V, F). The vertex set V is the set of all regions. The arc set F consists of two types of arcs:

- The arc $([s], [s'])$ may represent *meta-state transitions* in A, i.e., $s \to s'$.
- The arc $([s], [s'])$ may be a *time arc* and represent passage of time in the same meta-state. Formally
  - $s + \delta = s'$ for some $\delta \in \mathscr{R}^+$;
  - there is no $\dot{s}$ and $\dot{\delta} \in \mathscr{R}^+$, $0 < \dot{\delta} < \delta$, s.t. $[\dot{s}] \neq [s]$, $[\dot{s}] \neq [s']$, $s + \dot{\delta} = \dot{s}$, and $\dot{s} + \delta - \dot{\delta} = s'$.

For each $([s], [s'])$ in F, we let $\varepsilon([s], [s']) = \uparrow$ if going from s to s', the reading of κ increments from a noninteger to an integer; $\varepsilon([s], [s']) = \downarrow$ if going from s to s', the reading of κ increments from an integer to a noninteger; otherwise $\varepsilon([s], [s']) = 0$. Also $v \models fract(\kappa) = 0$ iff $s(\kappa)$ is an integer.

For convenience, we let $\langle \kappa \rangle [s]$ be the vertex, say $[s']$, in a CR-graph that agrees with $[s]$ in every aspect except that $s'(\kappa) = 0$.

A (finite or infinite) *path* $\langle v_1 v_2 \cdots \rangle$ in a CR-graph $G_A = (V, F)$ is a (finite or infinite) sequence of vertices such that for every $i \geqslant 1$, $(v_i, v_{i+1}) \in F$ if v_{i+1} exists. A *cycle* is a finite path $\langle v_1 v_2 \cdots v_m \rangle$ such that $m \geqslant 2$ and $v_1 = v_m$.

We now define the following notation to symbolize the time of a path. Given a path $\Gamma = \langle v_1 v_2 \cdots v_m \rangle$, we use $time(\Gamma)$ to denote the number of arcs (v_i, v_{i+1}) along Γ such that $\varepsilon(v_i, v_{i+1}) = \uparrow$. $time(\Gamma)$ is called the *time* of path Γ.

The models of PTCTL formulas are digressive computations along which a clock's reading will increase without bound in the case when it is not being reset infinitely often. These correspond to infinite paths, say $\langle v_1 v_2 \cdots \rangle$, in CR-graphs along which for infinitely many $i > 0$, $\varepsilon(v_i, v_{i+1}) = \uparrow$. A region satisfying $\exists \Box_{\geqslant 0} true$ can be checked by the reachability of a strongly connected component in the region graph with an $\uparrow$ arc in it.

5.2. Finite Characteristic of the Problem

Given a PTCTL formula ϕ and a path (cycle) $\Gamma = \langle v_1 v_2 \cdots v_m \rangle$, Γ is called a *ϕ-path* (ϕ-cycle) iff there is a parameter valuation $\mathscr{I}$ such that for each $1 \leqslant i < m$ and $v_i = [s_i]$, $s_i \models \phi^{\mathscr{I}}$.

We shall first give the intuition behind our labeling algorithm. The parametric timing analysis problem can be decomposed to the following basic problem:

> Given two vertices v, v' in a region graph and a property ϕ, what is the set of computation times of ϕ-paths from v to v'?

We shall show that any such computation time is always a derivable offset constant plus a multiple of the gcd of positive cycle times along some paths from v to v'. Once this is established, we shall use it to construct labeling functions for all modal formulas in PTCTL.

Any path from v to v' can always be decomposed into a simple path, say Γ, and a finite set, say H, of simple cycles. It is also observed that by repeating any of the simple cycles in H a few more times, we still get a path from v to v'. Thus the computation time of a path constructed from Γ and H can be represented as the sum of the time of Γ and a linear combination of the times of cycles in H. Note such a linear combination is always a nonnegative multiple of the gcd of the times of cycles in H. But the reverse is not true. Lemmas 2 and 3 step in at this point to show that when the nonnegative multiple is big enough, it is always possible to express it as a linear combination with nonnegative coefficients.

Lemma 3 actually establishes the intuition we made in Section 2. Namely, given two vertices v, v' in a region graph and a nonnegative constant Ψ as the gcd of

times of all positive cycles in some path from v to v', there exists $\Upsilon \in \mathcal{N}$ such that for any $i \in \mathcal{N}$, $\Upsilon + i\Psi$ describes a computation time from v to v'. Lemma 3 is established with the help of definition of path structures (which we call *cactus structure* in Definition 4) and Lemma 2.

What Lemma 2 does, given a set of positive integers $r_1, ..., r_m$ (presumably positive cycle times) and a linear combination c of $r_1, ..., r_m$ with $|c| \leqslant \text{lcm}(r_1, ..., r_m)$, is to find small constant bounds on $j_1, ..., j_m$ to make $c = \sum_{1 \leqslant i \leqslant m} j_i r_i$. Remember that all paths from v to v' are simple paths attached to sets of simple cycles. Once Lemma 2 is proven, it is easy to see that we need only cycle those simple cycles a constant number of times to make a path time composed of the simple path time and a linear combination of those cycle times with only positive coefficients. Such overhead cycle repetition leads to the derivation of Υ mentioned in the previous paragraph.

LEMMA 2. *Given a set of positive integers* $r_1, r_2, ..., r_m$, *for each integer* $0 \leqslant i < (\text{lcm}(r_1, r_2, ..., r_m))/(\gcd(r_1, r_2, ..., r_m))$, *there are integers* $j_1, ..., j_m$ *such that* $|j_h| \leqslant (\text{lcm}(r_1, ..., r_m))/r_h$, *for each* $1 \leqslant h \leqslant m$, *and* $\sum_{1 \leqslant h \leqslant m} j_h r_h = i \cdot \gcd(r_1, r_2, ..., r_m)$.

Proof. We first want to establish that for all i with $0 \leqslant i < (\text{lcm}(r_1, ..., r_m))/(\gcd(r_1, ..., r_m))$, $i \cdot \gcd(r_1, ..., r_m)$ can be represented as $\sum_{1 \leqslant h \leqslant m} j'_h r_h = i \cdot \gcd(r_1, ..., r_m)$, an integer linear combination (LC) of $r_1, ..., r_m$. This can be done by reversing Euclid's algorithm for gcd computation. After that, we want to show hat there are such integers $j_1, ..., j_m$ with $|j_h| \leqslant (\text{lcm}(r_1, ..., r_m))/r_h$ for each $1 \leqslant h \leqslant m$. Now assume $j'_h = ((\text{lcm}(r_1, ..., r_m))/r_h)\, k'_h + g_h$, where $0 \leqslant g_h < (\text{lcm}(r_1, ..., r_m))/r_h$, for each $1 \leqslant h \leqslant m$. By some further manipulation, we find

$$i \cdot \gcd(r_1, ..., r_m) = \sum_{1 \leqslant h \leqslant m} g_h r_h + \text{lcm}(r_1, ..., r_m) \sum_{1 \leqslant h \leqslant m} k'_h.$$

By dividing the two sides by $\text{lcm}(r_1, ..., r_m)$, we get

$$0 = \left\lfloor \frac{i \cdot \gcd(r_1, ..., r_m)}{\text{lcm}(r_1, ..., r_m)} \right\rfloor = \left\lfloor \frac{\sum_{1 \leqslant h \leqslant m} g_h r_h}{\text{lcm}(r_1, ..., r_m)} \right\rfloor + \sum_{1 \leqslant h \leqslant m} k'_h.$$

Since $0 \leqslant \lfloor (\sum_{1 \leqslant h \leqslant m} g_h r_h)/(\text{lcm}(r_1, ..., r_m)) \rfloor = -\sum_{1 \leqslant h \leqslant m} k'_h < m$, by letting $j_h = -((\text{lcm}(r_1, ..., r_m))/r_h) + g_h$ if $1 \leqslant h \leqslant |\sum_{1 \leqslant h \leqslant m} k'_h|$ and g_h otherwise, we find that the lemma is proven. ∎

Now the "bigness" of the nonnegative multiple comes into play. Here is the intuition. Suppose we are given a "big" $g \in \mathcal{N}$ such that

$$g \cdot \gcd(r_1, ..., r_m) \geqslant \sum_{1 \leqslant h \leqslant m} r_h + m \cdot \text{lcm}(r_1, ..., r_m).$$

Since for some $k \geqslant 0$ and $0 \leqslant i < (\text{lcm}(r_1, ..., r_m))/(\gcd(r_1, ..., r_m))$,

$$g \cdot \gcd(r_1, ..., r_m) = k \cdot \text{lcm}(r_1, ..., r_m) + \sum_{1 \leqslant h \leqslant m} r_h + m \cdot \text{lcm}(r_1, ..., r_m) + i \cdot \gcd(r_1, ..., r_m),$$

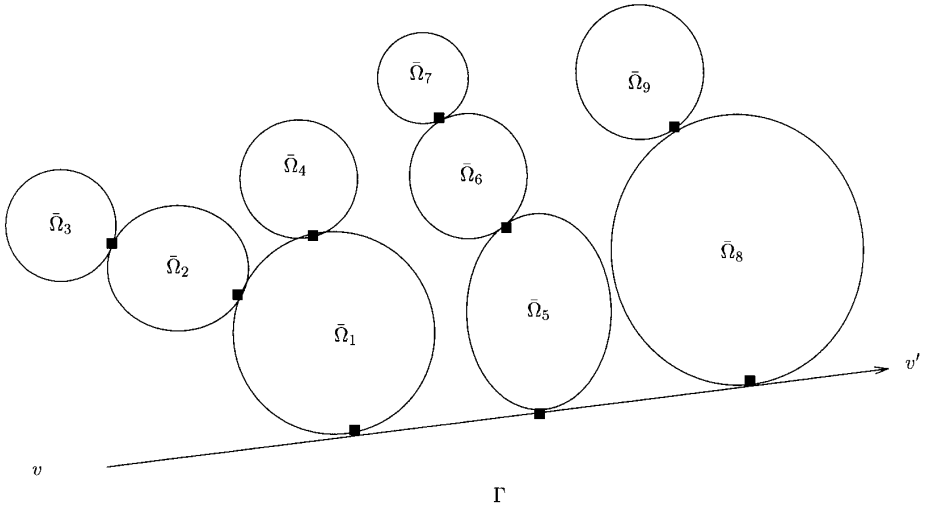

FIG. 3. A cactus structure.

according to Lemma 2,

$$g \cdot \gcd(r_1, ..., r_m) = k \cdot \mathrm{lcm}(r_1, ..., r_m) + \sum_{1 \leqslant h \leqslant m} r_h + m \cdot \mathrm{lcm}(r_1, ..., r_m) + \sum_{1 \leqslant h \leqslant m} j_h r_h$$

$$= \frac{\mathrm{lcm}(r_1, ..., r_m)}{r_1} \cdot k r_1 + \sum_{1 \leqslant h \leqslant m} \left(1 + \frac{\mathrm{lcm}(r_1, ..., r_h)}{r_h} + j_h\right) r_h.$$

Since $1 + ((\mathrm{lcm}(r_1, ..., r_h))/r_h) + j_h$ is a positive integer for each $1 \leqslant h \leqslant m$, a nonnegative multiple bigger than the "offset" of $\sum_{1 \leqslant h \leqslant m} r_h + m \cdot \mathrm{lcm}(r_1, ..., r_m)$ is guaranteed to be expressible as a linear combination of $r_1, ..., r_m$ with nonnegative coefficients. This intuition is made concrete and connected to our PTA problem by the following two definitions. The first defines the structure of linear combinations of cycle times in considering the computation times of paths between two vertices in CR-graph.

DEFINITION 4. *Cactus structure.* Given a PTA(A, ϕ), a simple path $\Gamma = \langle u_1 u_2 \cdots u_k \rangle$, and a finite set H of simple cycles in G_A, we call (Γ, H) a *cactus structure* iff for each $\Omega \in H$, there is a finite sequence $\Omega_1, ..., \Omega_m$, $m \geqslant 1$, of simple cycles in H such that $\Omega_1 = \Omega$ and

- given $\Omega_m = \langle v_1 \cdots v_n \rangle$, for some $1 \leqslant i \leqslant k$, $u_i = v_1$; and
- for each $1 \leqslant h < m$ with $\Omega_h = \langle v_1 v_2 \cdots v_m \rangle$ and $\Omega_{h+1} = \langle v'_1 v'_2 \cdots v'_n \rangle$, there is $1 \leqslant i \leqslant n$ msuch that $v'_i = v_1$.

An illustration of a cactus structure is given in Fig. 3. For example, for $\bar{\Omega}_7$, the sequence is $\bar{\Omega}_7 \bar{\Omega}_6 \bar{\Omega}_5$.

Given a set of nonnegative integers $r_1, ..., r_m$, we shall conveniently let $\gcd(r_1, ..., r_m)$ and $\mathrm{lcm}(r_1, ..., r_m)$ be respectively the *greatest common divisor* and the *least common multiple* of the "positive" elements in $r_1, ..., r_m$.

The following definition extracts the Υ and Ψ characteristics from a cactus structure in which some property ϕ is invariant except for the destination node.

DEFINITION 5. *Intrinsic conditional offset-period structure.* Given a PTCTL formula ϕ, two vertices v, v' in G_A, and a cactus structure (Γ, H) from v to v' in G_A, we call (Γ, H) an *intrinsic ϕ-offset-period structure* from v to v' iff there is a parameter valuation $\mathscr{I}$ of T_ϕ such that

- given $\Gamma = \langle u_1 \cdots u_n \rangle$, $\forall 1 \leqslant i < n$, $u_i \models \phi^{\mathscr{I}}$; and
- given $\langle v_1 \cdots v_n \rangle \in H$, $\forall 1 \leqslant i < n$, $v_i \models \phi^{\mathscr{I}}$.

Given the intrinsic ϕ-offset-period structure $(\Gamma, \{\Omega_1, ..., \Omega_m\})$ from v to v' in G with $r_h = time(\Omega_h)$ for each $1 \leqslant h \leqslant m$, we call the pair of numbers

$$\left(time(\Gamma) + \sum_{1 \leqslant h \leqslant m} r_h + m \cdot \mathrm{lcm}(r_1, ..., r_m), \gcd(r_1, ..., r_m) \right)$$

an *intrinsic ϕ-offset-period pair* from v to v' in G. For each intrinsic ϕ-offset-period pair (Υ, Ψ) from v to v', Υ and Ψ are called an *intrinsic ϕ-offset* and an *intrinsic ϕ-period*, respectively, from v to v'.

LEMMA 3. *Given a PTA(A, ϕ) and two vertices v, v' in G_A, for every $d \in \mathscr{N}$ such that*

$$d \geqslant \max\{\Upsilon \mid \Upsilon \text{ is an intrinsic } \phi\text{-offset from } v \text{ to } v'\}$$

there is a ϕ-path of time d from v to v' iff there are $i \geqslant 0$ and an intrinsic ϕ-offset-period pair (Υ, Ψ) from v to v' such that $d = \Upsilon + i \cdot \Psi$.

Proof. *Direction* $\Rightarrow$. By repetitively extracting simple cycles from the ϕ-path, we can decompose the path into a simple path Γ and a set H of simple cycles which together construct an intrinsic ϕ-offset-period structure. Let the relevant intrinsic ϕ-offset-period pair be (Υ, Ψ). Since they are decomposed from a ϕ-path, for some parameter valuation $\mathscr{I}$,

$$\forall \langle v_1 \cdots v_n \rangle \in H, \forall 1 \leqslant i < n \ (v_i \models \phi^{\mathscr{I}}).$$

Because of the composition of the path, for some $g \in \mathscr{N}$,

$$d = time(\Gamma) + g \cdot \Psi. \tag{1}$$

Also, because of the magnitude of d, we find that for some $c \in \mathscr{N}$,

$$d = time(\Gamma) + \sum_{\Omega \in H} time(\Omega) + c + |H| \cdot \mathrm{lcm}(time(\Omega) \mid \Omega \in H).^{1} \tag{2}$$

By subtracting Eqs. (1) and (2), we find that c must be a nonnegative multiple of Ψ and this direction of the lemma is proven.

[1] Given a set $H = \{\Omega_1, ..., \Omega_m\}$, $\mathrm{lcm}(time(\Omega) \mid \Omega \in H)$ means $\mathrm{lcm}(time(\Omega_1), ..., time(\Omega_m))$.

Direction $\Leftarrow$. Let $(\Gamma, \{\Omega_1, ..., \Omega_m\})$ and (Υ, Ψ) be the relevant intrinsic ϕ-offset-period structure and pair, respectively, with $r_h = time(\Omega_h)$ for each $1 \leqslant h \leqslant m$. For convenience, let $r_1 = \min\{r \mid r \in \{r_1, ..., r_m\};\ r \neq 0\}$. According to Lemma 2, there are integers $j_1, ..., j_m$ such that $|j_h| \leqslant ((\text{lcm}(r_1, ..., r_m))/r_h)$, for each $1 \leqslant h \leqslant m$, and

$$0 \leqslant \sum_{1 \leqslant h \leqslant m} j_h r_h = d - \Upsilon - \left\lfloor \frac{d-\Upsilon}{r_1} \right\rfloor r_1 < r_1 \leqslant \text{lcm}(r_1, ..., r_m).$$

At this step, we want to show that there exists such a ϕ-path with

- $e_1 = 1 + \lfloor (d-\Upsilon)/r_1 \rfloor + ((\text{lcm}(r_1, ..., r_m))/r_1) + j_1 > 0$ traversings of Ω_1, and
- $e_h = 1 + ((\text{lcm}(r_1, ..., r_m))/r_h) + j_h > 0$ traversings of Ω_h, for each $1 < h \leqslant m$, $r_h \neq 0$, respectively, and
- $e_h = 1$ traversing of Ω_h, for each $1 < h \leqslant m$, $r_h = 0$, respectively.

By visualizing e traversings of a cycle $\Omega = \langle v_1 \cdots v_n \rangle$ as a primary Ω together with $e-1$ replications of Ω conjoining at v_1, and constructing a dummy arc from the tail to the head of Γ, we actually get an *Eulerian circuit* problem instance [Knut73]. The directed graph is connected since it is a cactus structure and $\forall 1 \leqslant h \leqslant m$, $e_h \geqslant 1$. It is balanced since each cycle that is traversed adds both an incoming arc and an outgoing arc to each vertex along it. Thus according to Theorem G in [Knut73], the lemma is proven. ∎

The relation between the cactus structure and the intrinsic conditional offset-period structure can be illustrated by going through the region graph in Fig. 2 to construct the intrinsic conditional offset-period structure. As will be clear in our labeling algorithm in Table 2, the cactus structure that matters in Fig. 2 consists of exactly cycles Ω_1 and Ω_2. In this simple case, the simple path Γ is the trivial path of the only central p-region. Lemma 3 predicts that for every multiple d of $\gcd(2, 3) = 1$ with $d \geqslant 2 + 3 + 2\,\text{lcm}(2, 3) = 17$, d is a solution to σ in $\exists \Diamond_{=\sigma} p$.

Lemma 3 depicts the basic finite characteristic of our parametric timing analysis problem and thus prepares us to devise a labeling algorithm on CR-graphs for the problem.

5.3. Labeling Algorithm

In [ACD90, BCMDH90, CE81, and CES86], the model-checking algorithms label each vertex in the graphs with a set of temporal logic formulas true at the states represented by that vertex. Another way to look at this is that their labeling algorithms map pairs of vertices and temporal logic formulas to Boolean values. Our labeling algorithm can be viewed as an extension of theirs because it maps pairs of vertices and temporal logic formulas to a set of linear inequalities which we call *conditions*, with parameter variables as free variables. For convenience, we shall use the notation $L^{\phi}(v)$ for the conditions labeled on vertex v for PTCTL formula ϕ.

Given a path $\langle v_1 v_2 \cdots v_n \rangle$, it is always possible to decompose it into a simple path Γ and a set $\{\Omega_1, ..., \Omega_m\}$ of simple cycles. We call it a *slim* path if for each $1 \leqslant i \leqslant m$, $time(\Omega_i) = 0$ implies Ω_i is traversed at most once along Γ.

LEMMA 4. *Given two vertices v, v' and an $d \in \mathcal{N}$, there is a path from v to v' of time d iff there is a slim path from v to v' of time d.*

Proof. A path that is not slim can be reduced to a slim one by deleting duplicate zero cycles. ▮

Based on Lemmas 3 and 4, we devise a basic routine, $ptime^{\phi}_{\sim\theta}(\ )$ in Table 1, which, given a labeling function L^{ϕ}, two vertices v, v' in a CR-graph, and a requirement "$\sim\theta$" on the computation times of paths, returns a condition for the existence of a ϕ-path of computation time $\sim\theta$ from v to v'. Note how we use $L^{\phi}(\Gamma)$ and $\bigwedge_{\Omega \in H_\Gamma} L^{\phi}(\Omega)$ to transform existence conditions of a *true*-path to those of a ϕ-path. Also note that when $d \sim \theta$ is false for all $d \in \mathcal{N}$, $ptime^{\phi}_{\sim\theta}(v, v')$ will be evaluated to be false.

Our labeling algorithm is given in Table 2. The algorithm is given in a top-down recursive form for convenience. A bottom-up nonrecursive version may be more efficient. The following lemma establishes the correctness of our labeling algorithm.

LEMMA 5. *Given $PTA(A, \phi)$, a parameter valuation $\mathcal{I}$ for T_ϕ, and a vertex v in G_A, after executing $L^{\phi}(v)$ in our labeling algorithm, $\mathcal{I}$ satisfies $L^{\phi}(v)$ iff $v \models \phi^{\mathcal{I}}$.*

Proof. By structural induction on ϕ for both directions. We first assume that $\mathcal{I}$ satisfies $L^{\phi}(v)$ and want to show $s \models \phi^{\mathcal{I}}$. We prove this by induction on the structure of ϕ.

1. The case when $\phi = false$ is trivial.
2. The case when $\phi \in P$ is true according to the definition of clock regions.

TABLE 1

Conditions on Path Time for the Existence of ϕ-path

$ptime^{\phi}_{\sim\theta}(v, v')$ {
(1) let U be the set of simple paths from v to v';
(2) for each $\Gamma \in U$, {
(1) compute the set SC_Γ of vertices strongly connected to vertices in Γ;
(2) compute the set H_Γ of simple cycles in G_A made of vertices in SC_Γ;
(3) compute intrinsic *true*-offset-period pair $(\Upsilon_\Gamma, \Psi_\Gamma)$ for (Γ, H_Γ);
}
(3) let $\bar{\Upsilon} := \max\{\Upsilon_\Gamma \mid \Gamma \in U\}$ and $\eta := false$;
(4) for each $0 \leqslant d < \bar{\Upsilon}$ and each slim path of Γ of time d from v to v', $\eta := \eta \vee (d \sim \theta \wedge L^{\phi}(\Gamma))$
(5) for each $\Gamma \in U$, let $\eta := \eta \vee ((\exists i \geqslant \Upsilon_\Gamma + i \cdot \Psi_\Gamma \sim \theta)) \wedge L^{\phi}(\Gamma) \wedge \bigwedge_{\Omega \in H_\Gamma} L^{\phi}(\Omega))$;
(6) return η;
}

Note. Given a path $\Gamma = \langle v_1 \cdots v \rangle$, $L^{\phi}(\Gamma)$ is a shorthand for $\bigwedge_{1 \leqslant i < n} L^{\phi}(v_i)$.

TABLE 2

Labeling Algorithm

Label(A, ϕ) {
(1) construct the CR-graph $G_A = (V, F)$ for $PTA(A, \phi)$;
(2) for each $v \in V$, recursively compute $L^{\phi}(v)$;
}

$L^{\phi_i}(v)$ {
(1) if ϕ_i is *false*, $L^{false}(v) := false$;
(2) when $\phi_i \in P$, if $v \models \phi_i$, then $L^{\phi_i}(v) := true$, else $L^{\phi_i}(v) := false$;
(3) if ϕ_i is of the form $\phi_j \to \phi_k$, $L^{\phi_j \to \phi_k}(v) := L^{\phi_j}(v) \to L^{\phi_k}(v)$;
(4) if ϕ_i is of the form $\exists \Box_{\geq 0} \phi_j$, let $L^{\exists \Box_{\geq 0} \phi_j}(v)$ be $\bigvee_{u \in V} \left(ptime^{\phi_j}_{\geq 0}(\langle \kappa \rangle v, u) \wedge ptime^{\phi_j}_{>0}(u, u) \right)$.
(5) if ϕ_i is of the form $\exists \phi_j \mathcal{U}_{\geq \theta} \phi_k$, let $L^{\exists \phi_j \mathcal{U}_{\geq \theta} \phi_k}(v)$ be $\bigvee_{u \in V} \left(ptime^{\phi_j}_{\geq \theta}(\langle \kappa \rangle v, u) \wedge L^{\phi_k}(u) \wedge L^{\exists \Box_{\geq 0} \text{true}}(u) \right)$.
(6) the cases when ϕ_i is one of $\exists \phi_j \mathcal{U}_{\leq \theta} \phi_k$, $\exists \phi_j \mathcal{U}_{> \theta} \phi_k$, $\exists \phi_j \mathcal{U}_{< \theta} \phi_k$, and $\exists \phi_j \mathcal{U}_{= \theta} \phi_k$ are treated in ways similar to statement (5) and are left in table 3 which is appended at the end of the paper.
(7) if ϕ_i is of the form $\forall \phi_j \mathcal{U}_{\geq \theta} \phi_k$, let $L^{\forall \phi_j \mathcal{U}_{\geq \theta} \phi_k}(v)$ be

$$\neg \left(\begin{array}{l} L^{\exists \Diamond_{< \theta} \neg \phi_j}(\langle \kappa \rangle v) \ \vee \ \left(\theta = 0 \wedge \left(L^{\exists \Box_{\geq 0} \neg \phi_k}(\langle \kappa \rangle v) \vee L^{\exists (\neg \phi_k) \mathcal{U}_{\geq 0} \neg (\phi_j \vee \phi_k)}(\langle \kappa \rangle v) \right) \right) \\ \vee \ \left(\theta > 0 \wedge \bigvee_{u_1, u_2 \in V} \left(\begin{array}{l} ptime^{\phi_j}_{= \theta - 1}(\langle \kappa \rangle v, u_1) \wedge L^{\phi_j}(u_1) \wedge \epsilon(u_1, u_2) = \uparrow \\ \wedge \ \left(L^{\exists \Box_{\geq 0} \neg \phi_k}(u_2) \vee L^{\exists (\neg \phi_k) \mathcal{U}_{\geq 0} \neg (\phi_j \vee \phi_k)}(u_2) \right) \end{array} \right) \right) \end{array} \right)$$

(8) the cases when ϕ_i is one of $\forall \phi_j \mathcal{U}_{\leq \theta} \phi_k$, $\forall \phi_j \mathcal{U}_{> \theta} \phi_k$, $\forall \phi_j \mathcal{U}_{< \theta} \phi_k$, and $\forall \phi_j \mathcal{U}_{= \theta} \phi_k$ are treated in ways similar to statement (7) and are left in table 3 which is appended at the end of the paper.
}

3. Suppose ϕ is $\phi_j \to \phi_k$. According to our assumption, $\mathcal{I}$ satisfies $L^{\phi_j}(v) \to L^{\phi_k}(v)$. According to the inductive hypothesis, we then have $s \models \phi_j^{\mathcal{I}} \to s \models \phi_k^{\mathcal{I}}$, which in turn means that $s \models (\phi_j \to \phi_k)^{\mathcal{I}}$.

4. Suppose ϕ is $\exists \Box_{\geq 0} \phi_j$. According to statement (4) of Label(A, ϕ), the satisfaction of condition $L^{\exists \Box_{\geq 0} \phi_j}(v)$ by $\mathcal{I}$ says that there is a ϕ_j-cycle of positive time accessible from u through a ϕ_j-path from $\langle \kappa \rangle v$ to u. According to the construction of the CR-graph and our inductive hypothesis, this means starting at s in A, $\exists \Box_{\geq 0} \phi_j^{\mathcal{I}}$ is true, and the case is proven.

5. Suppose ϕ is $\exists \phi_j \mathcal{U}_{\geq \theta} \phi_k$. When $\mathcal{I}$ satisfies condition $L^{\exists \phi_j \mathcal{U}_{\geq \theta} \phi_k}(v)$, there exists a ϕ_j-path from $\langle \kappa \rangle v$ to u of time $\sim \mathcal{I}(\theta)$, ϕ_k is satisfied at u, and an $\exists \Box_{\geq 0} true$ is true at u. According to the construction of the CR-graph and our inductive hypothesis, this means $s \models \exists \phi_j^{\mathcal{I}} \mathcal{U}_{\geq \mathcal{I}(\theta)} \phi_k^{\mathcal{I}}$ and the case is proven.

6. The other cases of $\exists \mathcal{U}_{\sim \theta}$ can be proven similarly as in case 5.

7. Suppose ϕ is $\forall \phi_j \mathcal{U}_{\geq 0} \phi_k$. We work on the negation instead. $\forall \phi_j \mathcal{U}_{\geq 0} \phi_k$ is false exactly when one of the following three conditions happens:

— for some path, before θ time units, ϕ_j becomes false;

— for some path, for $\theta = 0$ and $\theta > 0$, respectively, after θ time units, ϕ_k is never true;

— for some path, for $\theta = 0$ and $\theta > 0$, respectively, after θ time units, ϕ_j becomes false before ϕ_k becomes true.

TABLE 3

Elaboration on Cases in Label(A, ϕ)

Elaboration on cases in Label(A, ϕ) {

(1) if ϕ_i is of the form $\exists\phi_j \mathcal{U}_{>\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{>\theta}\phi_k}(v)$ be

$$\bigvee_{u\in V}\left(\left(ptime^{\phi_j}_{>\theta}(\langle\kappa\rangle v, u) \vee \left(ptime^{\phi_j}_{=\theta}(\langle\kappa\rangle v, u) \wedge u \models fract(\kappa) \neq 0\right)\right) \wedge L^{\phi_k}(u) \wedge L^{\exists\Box_{\geq 0}\text{true}}(u)\right)$$

(2) if ϕ_i is of the form $\exists\phi_j \mathcal{U}_{\leq\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{\leq\theta}\phi_k}(v)$ be

$$\bigvee_{u\in V}\left(\left(ptime^{\phi_j}_{<\theta}(\langle\kappa\rangle v, u) \vee \left(ptime^{\phi_j}_{=\theta}(\langle\kappa\rangle v, u) \wedge u \models fract(\kappa) = 0\right)\right) \wedge L^{\phi_k}(u) \wedge L^{\exists\Box_{\geq 0}\text{true}}(u)\right)$$

(3) if ϕ_i is of the form $\exists\phi_j \mathcal{U}_{<\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{<\theta}\phi_k}(v)$ be $\bigvee_{u\in V}\left(ptime^{\phi_j}_{<\theta}(\langle\kappa\rangle v, u) \wedge L^{\phi_k}(u) \wedge L^{\exists\Box_{\geq 0}\text{true}}(u)\right)$

(4) if ϕ_i is of the form $\exists\phi_j \mathcal{U}_{=\theta}\phi_k$, let $L^{\exists\phi_j\mathcal{U}_{=\theta}\phi_k}(v)$ be

$$\bigvee_{u\in V}\left(ptime^{\phi_j}_{=\theta}(\langle\kappa\rangle v, u) \wedge u \models fract(\kappa) = 0 \wedge L^{\phi_k}(u) \wedge L^{\exists\Box_{\geq 0}\text{true}}(u)\right)$$

(5) if ϕ_i is of the form $\forall\phi_j \mathcal{U}_{>\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{>\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{l} L^{\exists\Diamond_{\leq\theta}\neg\phi_j}(\langle\kappa\rangle v) \\ \vee \bigvee_{u_1,u_2\in V}\left(ptime^{\phi_j}_{=\theta}(\langle\kappa\rangle v, u_1) \wedge L^{\phi_j}(u_1) \wedge \epsilon(u_1,u_2) = \downarrow \wedge \left(\begin{array}{l} L^{\exists\Box_{\geq 0}\neg\phi_k}(u_2) \\ \vee L^{\exists(\neg\phi_k)\mathcal{U}_{\geq 0}\neg(\phi_j\vee\phi_k)}(u_2)\end{array}\right)\right)\end{array}\right)$$

(6) if ϕ_i is of the form $\forall\phi_j \mathcal{U}_{\leq\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{\leq\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{ll} & \left(\bigvee_{u_1,u_2\in V}\left(ptime^{\neg\phi_k}_{=\theta}(\langle\kappa\rangle v, u_1) \wedge L^{\neg\phi_k}(u_1) \wedge \epsilon(u_1,u_2) = \downarrow \wedge L^{\exists\Box_{\geq 0}\text{true}}(u_2)\right)\right) \\ \vee & L^{\exists(\neg\phi_k)\mathcal{U}_{\leq\theta}\neg(\phi_j\vee\phi_k)}(\langle\kappa\rangle v)\end{array}\right)$$

(7) if ϕ_i is of the form $\forall\phi_j \mathcal{U}_{<\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{<\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{ll} & \left(\bigvee_{u_1,u_2\in V}\left(ptime^{\neg\phi_j}_{=\theta-1}(\langle\kappa\rangle v, u_1) \wedge L^{\neg\phi_k}(u_1) \wedge \epsilon(u_1,u_2) = \uparrow \wedge L^{\exists\Box_{\geq 0}\text{true}}(u_2)\right)\right) \\ \vee & L^{\exists(\neg\phi_k)\mathcal{U}_{<\theta}\neg(\phi_j\vee\phi_k)}(\langle\kappa\rangle v)\end{array}\right)$$

(8) If ϕ_i is of the form $\forall\phi_j \mathcal{U}_{=\theta}\phi_k$, let $L^{\forall\phi_j\mathcal{U}_{=\theta}\phi_k}(v)$ be

$$\neg\left(\begin{array}{l} L^{\exists\Diamond_{<\theta}\neg\phi_j}(\langle\kappa\rangle v) \\ \vee\left(\theta = 0 \wedge \left(\begin{array}{l} \bigvee_{u_1,u_2\in V}\left(ptime^{\neg\phi_k}_{=0}(\langle\kappa\rangle v, u_1) \wedge L^{\neg\phi_k}(u_1) \wedge \epsilon(u_1,u_2) = \downarrow \wedge L^{\exists\Box_{\geq 0}\text{true}}(u_2)\right)\right) \\ \vee L^{\exists(\neg\phi_k)\mathcal{U}_{=0}\neg(\phi_j\vee\phi_k)}(\langle\kappa\rangle v)\end{array}\right)\right) \\ \vee\bigvee_{u_1,u_2\in V}\left(\begin{array}{l} \theta > 0 \wedge ptime^{\phi_j}_{=\theta-1}(\langle\kappa\rangle v, u_1) \wedge L^{\phi_j}(u_1) \wedge \epsilon(u_1,u_2) = \uparrow \\ \wedge\left(\begin{array}{l} \bigvee_{u_3,u_4\in V}\left(ptime^{\neg\phi_k}_{=0}(u_2,u_3) \wedge L^{\neg\phi_k}(u_3) \wedge \epsilon(u_3,u_4) = \downarrow \wedge L^{\exists\Box_{\geq 0}\text{true}}(u_4)\right) \\ \vee L^{\exists(\neg\phi_k)\mathcal{U}_{=0}\neg(\phi_j\vee\phi_k)}(u_2)\end{array}\right)\end{array}\right)\end{array}\right)$$

}

The falsity of $L^{\forall\phi_j\mathcal{U}\phi_k}(v)$ leads to the satisfaction of the disjunction of these three conditions.

8. The other cases of $\forall\mathcal{U}_{\sim\theta}$ can be proven as in case 7.

We next assume that $s \models \phi^{\mathcal{I}}$ and we want to show that $\mathcal{I}$ satisfies $L^{\phi}(v)$. Again we do this by induction on the structure of ϕ.

1. The case when $\phi = false$ is trivial.

2. The case when $\phi \in P$ is true according to the definition of clock regions.

3. Suppose ϕ is $\phi_j \to \phi_k$. According to our assumption, $s \models \phi_j \to \phi_k$, which in turn means $s \models \phi_j \to s \models \phi_k$. According to the inductive hypothesis, we then have that $\mathcal{I}$ satisfies $L^{\phi_j}(v)$ implies that $\mathcal{I}$ satisfies $L^{\phi_k}(v)$. This in turn means that $\mathcal{I}$ satisfies $L^{\phi_j}(v) \to L^{\phi_k}(v)$.

4. Suppose ϕ is $\exists\Box_{\geqslant 0}\phi_j$. According to the semantics of $\exists\Box_{\geqslant 0}$, there is an s-run along which $\phi_j^{\mathcal{I}}$ is always true. Then for each $v = [s] \in V$, this is described by the formula constructed in statement (4) of Label(A, ϕ) which, according to our construction of the CR-graph and our inductive hypothesis, means that $\mathcal{I}$ satisfies $L^{\phi}(v)$.

5. Suppose ϕ is $\exists\phi_j \mathcal{U}_{\geqslant\theta}\phi_k$. When $s \models \phi^{\mathcal{I}}$ is true, there is an s-run through s_1 such that from s to s_1 it takes $\sim\mathcal{I}(\theta)$ time units, from s to just before s_1, $\phi_j^{\mathcal{I}}$ is

satisfied, and at $s_1\phi_k^{\mathscr{I}}$ is satisfied. This in turn means there is a path in CR-graph such that $\mathscr{I}$ satisfies $L^{\phi}(v)$ by our inductive hypothesis and CR-graph construction.

6. The other cases of $\exists\mathscr{U}_{\sim\theta}$ can be proven as in case 5.

7. Suppose ϕ is $\forall\phi_j\mathscr{U}_{\geqslant 0}\phi_k$. We work on the negation instead. $\forall\phi_j\mathscr{U}_{\geqslant 0}\phi_k$ is false exactly when one of the following three conditions happens:

— for some path, before θ time units, ϕ_j becomes false;

— for some path, for $\theta=0$ and $\theta>0$ respectively, after θ time units, ϕ_k is never true;

— for some path, for $\theta=0$ and $\theta>0$ respectively, after θ time units, ϕ_j becomes false before ϕ_k becomes true.

The satisfaction of the disjunction of these three conditions leads to the falsity of $L^{\forall\phi_j\mathscr{U}\phi_k}(v)$.

8. The other cases of $\forall\mathscr{U}_{\sim\theta}$ can be proven as in case 7.

This ends our proof. ∎

5.4. Complexity

We first analyze the time complexity of procedure *ptime*( ). The central part of *ptime*( ) deals with the enumeration of cycles and the calculation of the gcd's and lcm's of their positive cycle times. Suppose we are given a $PTA(A, \phi)$ with $A = (Q, q_0, P, C, \chi, E, \pi, \tau)$. For convenience, let $|A|$ be the size of A and $|\phi|$ be the size of ϕ in bits. According to our construction, the number of regions in G_A, denoted as $|G_A|$, is at most $3\,|Q|\cdot(K_A+1)^C\cdot(|C|+1)!$, where the coefficient 3 and the constant $+1$ reflect the introduction of the ticking indicator κ. The number of simple cycles in G_A is then $O((|G_A|+1)!)$, where the $+1$ reflects the insertion of an artificial token to mark the end of cycle node sequence in the permutation. Each simple cycle has at most $|G_A|$ vertices. Thus, to enumerate the simple cycles and their cycle times in the maximal strongly connected components, it takes time at most $O(|G_A|((|G_A|+1)!))$.

Given two integers m and n, their gcd and lcm can both be calculated in time complexity $O(\log(mn))$ under the assumption of constant-time multiplication and division. Since the corresponding gcd and lcm of the $O((|G_A|+1)!)$ cycle times should be no greater than $|G_A|$ and $|G_A|^{O((|G_A|+1)!)}$ respectively, they can also be computed in $O((|G_A|+1)!)$ iterations with total time complexity

$$O((|G_A|+1)!\,\log(|G_A|\cdot|G_A|^{(|G_A|)!})) = O((|G_A|+1)!\,(|G_A|)!\,\log|G_A|).$$

Summing up the two component times for cycle enumeration and gcd, lcm calculation, we conclude that the time complexity for *ptime*( ) is

$$\begin{aligned} &O(|G_A|((|G_A|+1)!)+(|G_A|+1)!\,(|G_A|)!\,\log|G_A|)\\ &\quad = O((|G_A|+1)!\,(|G_A|)!\,\log|G_A|). \end{aligned}$$

We now analyze the complexity of our labeling procedure. In Table 2, procedure $L^{\phi_i}(\,)$ invokes *ptime*( ) at most $|G_A|^4$ times while Label(A, ϕ) invokes $L^{\phi_i}(\,)$ at most

$|G_A|\,|\phi|$ times. With some simplification, we find that the time complexity of procedure Label( ) is polynomial in $|G_A|$ and $(|G_A|+1)!$, which means that the procedure Label( ) is in time $|\phi| \cdot 2^{2^{O(|A|)}}$, since polynomials of double exponentialities are still double exponentialities.

In summary, we find that our algorithm for parametric timing analysis problem is of double-exponential time complexity. Since the model-checking problem defined in [ACD90] can be viewed as a special case of our parametric timing analysis problem, we find that parametric timing analysis problem is at least PSPACE-hard. Thus, there is a gap here which needs more research effort to close.

Finally, the PTCTL satisfiability problem is undecidable since it is no easier than the TCTL satisfiability problem [ACD90].

5.5. *Finding a Solution of the PTA Condition*

There is a parameter valuation $\mathscr{I}$ for T_ϕ making A a model of $\phi^{\mathscr{I}}$ iff $\bigwedge_{v \in V;\, v \models \chi(q_0)} L^\phi(v)$, which we call the *intrinsic function* of $PTA(A, \phi)$, is satisfiable. A parameter valuation $\mathscr{I}$ for T_ϕ which makes $\bigwedge_{v \in V;\, v = \chi(q_0)} L^\phi(v)$ true is called a *solution* to $PTA(A, \phi)$. We adopt a straightforward approach to solving the intrinsic functions. By carrying out the following three steps, all intrinsic functions can be simplified to Boolean logic formulas with literals restricted to one of the four forms of *true*, *false*, $\sigma \sim c$, $\exists i(\sigma = d + i\Delta)$, where σ is a parameter variable and c, d, Δ are nonnegative integer constants.

- Atoms of the forms $\varepsilon(v, v') = \uparrow$, $\varepsilon(v, v') = \downarrow$, $\langle \kappa \rangle v = u_1$, $u \models fract(\kappa) = 0$, $c \sim d$, or $\exists i(c \sim d + i\Delta)$ can be evaluated to specific Boolean values.
- $\exists i(\sigma \geqslant d + i\Delta)$ and $\exists i(\sigma > d + i\Delta)$ are equivalent to $\sigma \geqslant d$ and $\sigma > d$, respectively. $\exists i(\sigma \leqslant d + i\Delta)$ and $\exists i(\sigma < d + i\Delta)$ are both equivalent to *true*.
- $\neg \exists i(\sigma = d + i\Delta)$ is equivalent to $(\bigvee_{0 \leqslant j < d} \sigma = j) \vee \bigvee_{0 < j < \Delta} \exists i(\sigma = d + j + i\Delta)$.

By simple induction on the structure of ϕ and case analysis on the construction of L^ϕ, it can be shown that these three steps indeed simplify intrinsic functions as claimed.

After the simplification, since there is only existential quantification, we can use standard techniques for linear equations to find a solution, if there is any.

6. CONCLUSION

With the success of CTL-based techniques in automatic verification for computer systems [Bryant86, BCMDH90, HNSY92], we feel hopeful that the insight and techniques used in this paper can be further applied to help verify reactive systems in a more natural and productive way. Especially, we demonstrate that for the model-checking problem, the region graph construction can be independent of the timing constants used in specification formula. Thus, the state-space explosion in verifying real-time systems can be further contained.

ACKNOWLEDGMENTS

The author thanks his colleague, Dr. Ming-Tat Ko, who showed how to prove the special case of Lemma 2 with $i = 1$ and $m = 2$. Also, special thanks to Dr. Churn-Jung Liau for his careful reading of the paper. Finally, the author thanks the reviewers for LICS'95 and this journal for all their suggestions and comments, which have greatly improved the presentation of the work.

Received March 3, 1995; final manuscript received September 2, 1996

REFERENCES

[ACD90] Alur, R., Courcoubetis, C., and Dill, D. L. (1990), Model-checking in dense real-time, *Inform. Comput.* **104**, 2–34.

[AFH91] Alur, R., Feder, T., and Henzinger, T. A. (1991), The benefits of relaxing punctuality, *in* "Proceedings, 10th ACM PODC."

[AH90] Alur, R., and Henzinger, T. A. (1990), Real-time logics: Complexity and expressiveness, *in* "Proceedings, 5th IEEE LICS."

[AHV93] Alur, R., Henzinger, T. A., and Vardi, M. Y. (1993), Parametric real-time reasoning, *in* "Proceedings, 25th ACM STOC," pp. 592–601.

[Bryant86] Bryant, R. E. (1986), Graph-based algorithms for Boolean function manipulation, *IEEE Trans. Comput.* **C-35** (8).

[BCMDH90] Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., and Hwang, L. J. (1990), Symbolic model checking: 10^{20} states and beyond, *in* "Proceedings, 5th IEEE LICS."

[CE81] Clarke, E., and Emerson, E. A. (1981), Design and synthesis of synchronization skeletons using branching-time temporal logic, *in* "Proceedings, Workshop on Logic of Programs," Lecture Notes in Computer Science, Vol. 131, Springer-Verlag, Berlin/New York.

[CES86] Clarke, E., Emerson, E. A., and Sistla A. P. (1986), Automatic verification of finite-state concurrent systems using temporal logic specifications, *ACM Trans. Programming Languages Systems* **8** (2), 244–263.

[CY92] Courcoubetis, C., and Yannakakis, M. (1992), Minimum and maximum delay problems in real-time systems, "Formal Methods in System Design," Vol. 1, 385–415, Kluwer Academic, Dordrecht/Norwell, MA; also *in* "Proceedings, 3rd CAV," LNCS 575, Springer-Verlag, Berlin/New York, 1991.

[EMSS90] Emerson, E. A., Mok, A. K., Sistla, A. P., and Srinivasan, J. (1990), Quantitative temporal reasoning, *in* "Proceedings, 2nd CAV," LNCS 531, Springer-Verlag, Berlin/New York.

[HLP90] Harel, E., Lichtenstein, O., Pnueli, A. (1990), Explicit clock temporal logic, *in* "Proceedings, 5th IEEE LICS."

[HNSY92] Henzinger, T. A., Nicollin, X., Sifakis, J., Yovine, S. (1992), Symbolic model checking for real-time systems, *in* "Proceedings, 7th IEEE LICS."

[Lewis90] Lewis, H. R. (1990), A logic of concrete time intervals, *in* "Proceedings, 5th IEEE LICS."

[Knuth73] Knuth, D. E. (1973), "The Art of Computer Programming," 2nd ed., Vol. 1, Addison–Wesley, pp. 373–374.

[McMillan92] McMillan, K. L. (1992), "Symbolic Model Checking : An Approach to the State Explosion Problem," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University.

[Pnueli77] Pnueli, A. (1977), The temporal logic of programs, *in* "Proceedings, 18th Annual IEEE–CS FOCS," pp. 45–57.

[WME93a] Wang, F., Mok, A., and Emerson, E. A. (1993), Symbolic model checking for distributed real-time systems, *in* "Proceedings, 1st Formal Methods Europe Symposium, Odense, Denmark," LNCS 670, Springer-Verlag, Berlin/New York.

[WME93b] Wang, F., Mok, A. K., and Emerson, E. A. (1993), Real-time distributed system specification and verification in APTL, *ACM Transl. Software Engrg. Methodol.* **2** (4), 346–378; also *in* "Proceedings, 14th ACM ICSE," 1992.